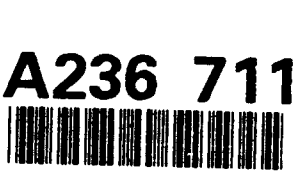

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

THE PORTING OF A MAINFRAME-DEPENDENT ANTENNA MODELING PROGRAM (NEC-3) TO A 32-BIT PERSONAL COMPUTER

by

JAMES J. WRIGHT

June, 1990

Thesis Advisor: Richard W. Adler

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No 0704-0188

1a REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION: Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable): EC	7a NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000
8a NAME OF FUNDING / SPONSORING ORGANIZATION; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification): THE PORTING OF A MAINFRAME-DEPENDENT MODELING PROGRAM (NEC-3) TO A 32-BIT PERSONAL COMPUTER

12 PERSONAL AUTHOR(S): WRIGHT, James J.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1990 June	98

16 SUPPLEMENTARY NOTATION: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Antenna Modeling; Numerical Electromagnetics Code (NEC-3); FORTRAN

19 ABSTRACT (Continue on reverse if necessary and identify by block number) The purpose of this thesis was to demonstrate the feasibility of porting a large mainframe-dependent scientific FORTRAN program, specifically the Numerical Electromagnetics Code (NEC-3) to a 32-bit personal computer. Two systems, an AST Premium 386/33 with both Intel 80387 and Weitek 23167 math co-processors and a Definicon DSI-780 using a Motorola 68020 CPU and 68881 math co-processor, were used with several 32-bit FORTRAN 77 compilers.

Results show that when NEC-3 was promoted to full double precision, complete accuracy was maintained while suffering only a 12% increase in execution time over single precision. Testing also revealed that the double precision Weitek version is 30% faster than the 80387 version. Some small inaccuracies remain; however, the same results were obtained by both the Naval Postgraduate School mainframe's new IBM VS2 FORTRAN 77 compiler and the personal computer FORTRAN 77 compilers. This indicates that the "bug" is in the NEC-3 code, vice being a hardware or FORTRAN 77 compiler problem.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED

22a NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code)	22c OFFICE SYMBOL
ADLER, Richard W.	408-646-2352	EC/Ab

DD Form 1473, JUN 86 Previous editions solete S/N 0102-L -6603

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

Approved for public release, distribution is unlimited

The Porting of a Mainframe-dependent Antenna Modeling Program (NEC-3) to a 32-Bit Personal Computer

by

James J. Wright
Lieutenant, United States Navy
B.S., United States Naval Academy, 1983

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL

June 1990

Author: James J. Wright

Approved by: Richard W. Adler, Thesis Advisor

Michael A. Morgan, Second Reader

John P. Powers, Chairman,
Department of Electrical and Computer Engineering

ABSTRACT

The purpose of this research was to demonstrate the feasibility of porting a large mainframe-dependent scientific FORTRAN program, specifically the Numerical Electromagnetics Code (NEC-3) to a 32-bit personal computer. Two systems, an AST Premium 386/33 with both Intel 80387 and Weitek w3167 math co-processors and a Definicon DSI-780 using a Motorola 68020 CPU and 68881 math co-processor, were used with several 32-bit FORTRAN 77 compilers.

Results show that when NEC-3 was promoted to full double precision, complete accuracy was maintained while suffering only a 12% increase in execution time over single precision. Testing also revealed that the double precision Weitek version is 30% faster than the 80387 version. Some small inaccuracies remain; however, the same results were obtained by both the Naval Postgraduate School mainframe's new IBM VS2 FORTRAN 77 compiler and the personal computer FORTRAN 77 compilers. This indicates that the "bug" is in the NEC-3 code, vice being a hardware or FORTRAN 77 compiler problem.

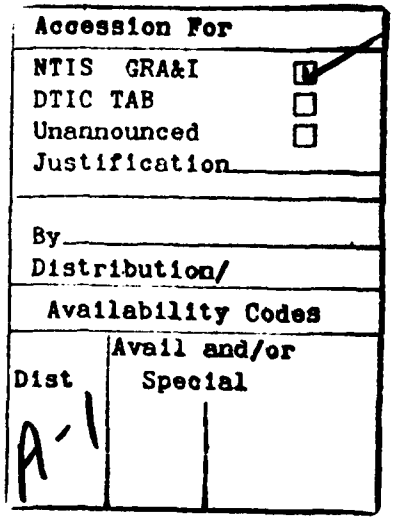

TABLE OF CONTENTS

I. INTRODUCTION 1

A. PURPOSE 1

B. BACKGROUND 2

1. Mainframe Computer Versions 2

2. Personal Computer Versions 3

C. CONTENTS 5

II. PROGRAMMING ENVIRONMENT 7

A. HARDWARE 7

1. AST Premium 386/33 7

2. Definicon DSI-780 Co-processor Board 8

B. FORTRAN COMPILERS 8

1. Lahey F77L-EM/32 FORTRAN Compiler (v.2.1) 9

2. MicroWay NDP FORTRAN-386 Compiler (v.1.4E) 10

3. SVS FORTRAN-386 Compiler (v.2.8) 11

4. SVS Definicon FORTRAN Compiler (v.2.8) 11

III. PROGRAM SOURCE CODE 13

A. THE FORTRAN PROGRAMMING LANGUAGE 13

1. Major Differences Between FORTRAN 77 and FORTRAN 66 14

a. Structured Branching Statements 14

b. Character Data Type 14

c. DO Loop Changes 14

d. List Directed Input and Output 15

e. Expressions 15

f. Compile-time Constants 15

g. IMPLICIT Type Declaration 15

h. Generic Intrinsic Functions 16

i. Subprogram Reference 16

j. Array Bounds 16

k. Computed GOTO Default 16

l. INPUT/OUTPUT Statements 17

m. SAVE Statement 17

n. FORTRAN Character Set/Comment Lines 17

2. Portability of FORTRAN Programs 18

a. Language Elements 19

b. Expressions and Assignments 21

c. Control Statements 22

d. Specification Statements 23

e. Program Units 24

f. Input/Output 27

B. OVERVIEW OF THE NEC-3 FORTRAN PROGRAM 28

1. Theory 28

2. Program Organization 30

C. MODIFICATIONS INCORPORATED INTO THE NEC-3 FORTRAN PROGRAM 30

1. NEC-3 Modifications 30

a. SAVE Statements 30

b. Common Block Alignment 31

c. Full Double Precision 31

d. Generic Functions 32

e. FORTRAN 77 Standard Array Passing 32

f. Output Statements 32

g. External Functions 33

h. Subroutine N.names 33

i. INTEGER*4, REAL*8, and COMPLEX*16 33

j. Arrays of Mixed Data Types 33

k. EQUIVALENCE Statements with Mixed Data Types . . . 34

l. File I/O Modifications 35

m. Compiler-specific Subroutines 36
2. SOMNTX Modifications 36

IV. RESULTS 37
A. DIPOLE EXAMPLES 37
B. BASIC ABOVE GROUND PROBLEMS 42
1. Rhombic Antenna 43
2. NEC User's Guide Problems 43
C. GROUND PROBLEMS 52

V. CONCLUSIONS 58

APPENDIX A (LAHEY F77L3 FORTRAN COMPILER) 60

APPENDIX B (MICROWAY NDP FORTRAN-386 COMPILER) 62

APPENDIX C (SVS FORTRAN COMPILERS) 65

APPENDIX D (TIMER AND VERSION SUBROUTINES) 67

APPENDIX E (NEC TEST PROBLEMS) 71

LIST OF REFERENCES 81

INITIAL DISTRIBUTION LIST 83

LIST OF TABLES

TABLE 4.1 SINGLE VERSUS DOUBLE PRECISION 38

TABLE 4.2 49-SEGMENT DIPOLE 39

TABLE 4.3 99-SEGMENT DIPOLE 39

TABLE 4.4 199-SEGMENT DIPOLE 40

TABLE 4.5 299-SEGMENT DIPOLE 40

TABLE 4.6 EXECUTION TIMES OF 299-SEGMENT VS. 301-SEGMENT DIPOLES 41

TABLE 4.7 RHOMBIC ANTENNA HORIZONTALLY POLARIZED .. 43

TABLE 4.8 CENTER FED LINEAR ANTENNA 44

TABLE 4.9 CENTER FED LINEAR ANTENNA, VARIABLE FREQUENCY 45

TABLE 4.10 VERTICAL HALF-WAVE DIPOLE OVER GROUND 46

TABLE 4.11 T-ANTENNA ON A BOX OVER PERFECT GROUND ... 47

TABLE 4.12 12 ELEMENT LOG PERIODIC ANTENNA IN FREE SPACE 47

TABLE 4.13 CYLINDER WITH ATTACHED WIRE 48

TABLE 4.14 SCATTERING BY A WIRE 49

TABLE 4.15 STICK MODEL OF AN AIRCRAFT IN FREE SPACE 50

TABLE 4.16 BISTATIC SCATTERING BY A SPHERE 51

TABLE 4.17 MONOPOLE OVER A RADIAL WIRE SCREEN OVER FINITE GROUND 53

TABLE 4.18 DIPOLE ANTENNA MOVING CLOSER TO GROUND .. 54

TABLE 4.19 DIPOLE IN DIELECTRIC AND LOSSY MEDIA 55

TABLE 4.20 MONOPOLE ON A GROUND STAKE 56

TABLE 4.21 MONOPOLE ON A 6-WIRE RADIAL GROUND SCREEN 57

I. INTRODUCTION

A. PURPOSE

The Numerical Electromagnetics Code version-3 (NEC-3) advanced antenna modeling program was first implemented on the Naval Postgraduate School IBM 3033AP mainframe computer in 1983. Since then, NEC-3 has proven to ' e an invaluable tool in antenna design, particularly in the high interest area of buried antennas. The use of a personal computer version of NEC-3 will greatly benefit researchers by freeing them from the constraints of the mainframe computer.

The principle objective of the research was the evaluation of the performance of a large mainframe-dependent FORTRAN program, specifically NEC-3, which had been ported to a fast 32-bit personal computer. Recent advances in microcomputer architecture, semiconductor memory, and hard disk drive performance now make it practical to implement such large scale scientific FORTRAN programs on a pe-sonal computer. Technically, the mainframe is still faster than most 32-bit personal computers, although this speed advantage is lost if more than a few users are on the system. The relatively inexpensive cost, reliability, portability, and user friendliness have made the 32-bit personal computer the ideal computer system for scientific programming and computation.

B. BACKGROUND

1. Mainframe Computer Versions

The NEC-3 is a user-oriented FORTRAN program for the analysis of the electromagnetic response of antennas and other metal structures. NEC-3 is the latest version of a series of electromagnetic FORTRAN programs, each of which improved upon its predecessors. The first program was BRACT, developed by MBAssociates in San Ramon, California in 1970, under the funding of the Air Force Space and Missiles Systems Organization. BRACT was specialized to scattering by arbitrary thin-wire configurations. In 1974, the Antenna Modeling Program (AMP) was developed by MBAssociates with funding from the Naval Research Laboratory, Naval Ships Engineering Center, U.S. Army ECOM/Communications Systems, U.S. Army Strategic Communications Command, and the Rome Air Development Center. AMP improved upon BRACT in several aspects by adding the capability of modeling a structure over a ground plane and allowing the use of file storage, thereby increasing the maximum structure size that could be modeled. AMP-2, introduced in 1975, included a simplified approximation for large interaction distances which helped reduce run time for large structures. [Ref. 1:p. 1]

NEC-1 was developed at the Lawrence Livermore National Laboratory (LLNL) under the sponsorship of the Naval Ocean Systems Center (NOSC) in 1977. NEC-1 added to AMP2 a more accurate current expansion along wires and at wire junctions, and an option in the wire modeling technique for greater accuracy for

thick wires. A new model for a voltage source was included along with several other modifications made for increased accuracy and efficiency. NEC-2 was introduced in 1980, with the added capability of performing the Numerical Green's Function for partitioned-matrix solutions and a treatment for lossy grounds that is accurate for antennas very close to the ground surface. In 1983, the current version, NEC-3, was released. This version has the added capability of modeling buried antennas and wires penetrating a lossy media. [Ref. 1:p. 2]

2. Personal Computer Versions

The personal computer's relatively low cost and increasing capabilities have made it the focus of several efforts to implement a working version of NEC for a personal computer.

The NOSC developed a Mini-Numerical Electromagnetics Code (MININEC) in 1982. This program was written in BASIC and included many, but not all of the capabilities of the full NEC program. MININEC was also limited to modeling only electrically small wire structures by the amount of memory available (640 KB) in the personal computer. [Ref. 2:p. 1]

Stephan Lamont of the Naval Postgraduate School (NPS) was the first known researcher to implement the full version of the NEC-3 program on small computer systems in 1986. The three systems that he used were:

- An IBM PC-AT, as an example of a commonly available "off the shelf" microcomputer.

- An IBM PC-AT/370, as a middle-to-high end micro-mainframe
- An IBM RT PC, as an example of a high-performance, high-function workstation embodying a novel architecture and with the capabilities outside the range of personal computers. [Ref. 3:pp. 1-2]

The results of his research were somewhat promising and showed that for the first time it was possible to run the full NEC-3 program on a small "personal" system. Both the IBM RT PC and IBM PC AT/370 executed without any problems; the IBM PC AT, however, suffered from several problems which are summarized as follows:

- The IBM Professional FORTRAN did not support the DOUBLE PRECISION COMPLEX data type, thus requiring double precision complex algorithms to be coded in two subroutines to provide the required accuracy.
- When processing an out-of-core solution, a record length error frequently occurred, requiring the addition of a run-time parameter specifying the maximum record length allowable.
- Out-of-core solutions did not produce correct results.
- The speed of the IBM PC-AT was between 2 and 10 times slower than that of the IBM RT PC or IBM PC-AT/370. [Ref. 3:pp. 3-6]

Captain Timothy O'Hara, U.S. Army, implemented the full NEC-3 program on three 32-bit PC systems in 1988:

- a COMPAQ 386/20 with both Intel 80387 and Weitek w1167 math co-processors,
- an IBM RT PC with a National Semiconductor (NS) 32091 floating point accelerator, and

- a Definicon DSI-780 co-processor board with a Motorola 68020/20 and a Motorola 68881 math co-processor.

All three systems had large in-core memories, with 4 Megabytes each. The 32-bit MicroWay FORTRAN compiler used on the COMPAQ 386/20 implemented extended memory via a DOS Extender made by Phar Lap Tools. [Ref. 4:pp. 3-4]

The results of O'Hara's research showed that although both the Definicon and IBM RT PC systems generated accurate data for simple problems, their execution times were not fast enough to justify their use for large-scale problems. The COMPAQ 386/20, however, was fast enough but had several very serious accuracy problems. When the Intel 387 math chip was used, the antenna input impedances became stochastic because of a COMMON block alignment problem. When the Weitek w1167 was used, the results became more inaccurate as the number of unknowns in the problem increased. The Weitek inaccuracy problem was attributed to the small single precision mantissa of the Weitek chip, 23 bits versus 53 bits in double precision. [Ref. 4:pp. 33-34]

C. CONTENTS

Chapter II contains a detailed description of the AST Premium 386/33 personal computer, the Definicon DSI-780 co-processor board, and the FORTRAN compilers used by each system.

Chapter III contains a description of past and current FORTRAN language standards, their impact upon the NEC-3 program, and the specific changes that were made to NEC-3 during the research.

Chapter IV contains a description of the testing of the latest personal computer version of NEC-3 and the results.

Chapter V contains conclusions and recommendations.

II. PROGRAMMING ENVIRONMENT

Two separate 32-bit computer systems were utilized for this research, an AST Research Premium 386/33 MS-DOS compatible system and a Definicon DSI-780 co-processor board. Three different FORTRAN compilers were used on the MS-DOS system in order to provide a ready means of completing the project in the event that any one compiler was unsuccessful in running the NEC-3 program. Only one FORTRAN compiler was available for the Definicon DSI-780 co-processor board, and the most recent version of that compiler was used.

A. HARDWARE

1. AST Premium 386/33

The AST Premium 386/33 System is a very high performance MS-DOS system using an Intel 80386 microprocessor running at 33 MHz. The system has 4 MB of 32-bit memory on the motherboard, 8 MB of 32-bit memory installed on an AST memory expansion board, and 32 KB of cache memory. Both Intel 80387 running at 33 MHz and Weitek w3167 floating point co-processors are installed on the motherboard. Hard disk storage is provided by a fast EDSI 100 MB hard disk, and the Microsoft SMARTDRV.SYS disk cache routine is used to improve disk

performance throughput. The Norton Utilities computing index (CI) for this system relative to the IBM/XT is 38.8. [Ref. 5:p. xi]

2. Definicon DSI-780 Co-processor Board

The Definicon DSI-780 is a 32-bit microprocessor, running at 20 MHz, a 32-bit data bus, an MC 68881 floating point math co-processor, and 4 MB of 32 bit dynamic memory (RAM). The board interfaces the MS-DOS operating system of PC-AT computers via a 64 KB window located at D000H. [Ref. 6:pp. 1-3]

B. FORTRAN COMPILERS

The three MS-DOS FORTRAN compilers selected for this research were chosen because they support the use of extended memory and the DOUBLE PRECISION COMPLEX data type. The extended memory feature was a requirement because of the large program size and in-core data storage requirements of the NEC-3 program. The DOUBLE PRECISION COMPLEX feature was required to produce a full double precision version of NEC-3. This was desirable because the IBM mainframe version was promoted to full double precision at compile time, and the single precision version run by O'Hara had some precision related accuracy problems.

The only FORTRAN compiler available for the Definicon DSI-780 co-processor board was the SVS FORTRAN compiler. The newest version, v.2.8, was used as it now supports the DOUBLE PRECISION COMPLEX data type. Memory

access on the Definicon DSI-780 was not a problem since it was not affected by the MS-DOS memory limit.

1. Lahey F77L-EM/32 FORTRAN Compiler (v.2.1)

The F77L-EM/32 32-bit FORTRAN compiler v.2.1 is produced by Lahey Computer Systems, Inc. of Incline Village, Nevada. This compiler fully supports the Intel 80386 microprocessor, Intel 80387 math co-processor, and the Weitek w3167 math co-processor. Extended memory support is provided via a transparent DOS Extender which requires no special action by the program to use extended memory. [Ref. 7:p. i]

The F77L compiler is a true native code compiler that compiles directly to 80386 machine code without going through an assembly language first. The compiler code is linked into an executable file with an (.EXP) extension by the Lahey linker (L32). The executable file requires the Lahey loader (UP) in order to execute properly in extended memory. Appendix A contains a complete listing of all switches and options available with the Lahey F77L FORTRAN compiler. Of special note is the "/Q1" switch. This switch forces the compiler to limit the math co-processor stack to eight registers, which is the physical limit of the math co-processor. The default case is for this switch to be turned off, which allows the math co-processor stack to grow beyond eight registers via an exception trap to the runtime library scheme. When this happens while executing NEC-3, the results were completely inaccurate and unusable. It is unclear why an unlimited math

co-processor stack would be useful to anyone. The documentation only indicated that the switch rarely causes any problems, which in this case includes the AST 386/33. NEC-3 executed properly when the switch was turned on. [Ref. 7:pp. A1-A8]

2. MicroWay NDP FORTRAN-386 Compiler (v.1.4E)

The 32-bit NDP FORTRAN-386 compiler v.1.4E is produced by MicroWay, Inc., of Kingston, Massachusetts. Full support is provided for the Intel 80386 microprocessor, Intel 80287 and 80387 math co-processors, and the Weitek w3167 math co-processor. Full access to extended memory is provided with the Phar Lap Tools DOS Extender. The DOS extender, however, is not part of the NDP FORTRAN-386 package and must be purchased separately in order to produce executable files. The DOS extender is transparent to the programmer, and extended memory is fully available to the program at execution time. [Ref. 8:pp. 1.1-1.2]

NDP FORTRAN-386 compiles to an intermediate assembly language code, which is then assembled by the Phar Lap Tools assembler, ASM386, into object code. The Phar Lap Tools linker, LINK386, produces executable (.EXP) file from the object file. The Phar Lap Tools loader, LOAD386, is required to load the executable file into extended memory. Appendix B contains a complete list of all compiler switches and options. [Ref. 8:pp. 1.5-1.12]

3. SVS FORTRAN-386 Compiler (v.2.8)

The SVS FORTRAN-386 compiler v.2.8 was distributed by Science Applications International Corporation (SAIC) of Los Altos, California. The compiler is produced by Silicon Valley Software (SVS) of Cupertino, California. The linker is produced by Phar Lap Tools and the loading executive manager is provided by Intelligent Graphics Corporation (IGC) of Santa Clara, California. Full support for extended memory is provided by the VM/RUN loading execution manager. All DOS extended memory is available to the program at execution time. [Ref. 9:p. 1.1]

SVS FORTRAN-386 is a three pass compiler which uses three separate programs to compile the source file into an object file (.OBJ). The object file is then linked into an executable file by the Phar Lap Tools Linker (LINK386). The loading execution manager, VM/RUN, is required to execute the file in extended memory. Appendix C contains a list of all compiler switches and options. [Ref. 9:pp. 3.1-3.2]

4. SVS Definicon FORTRAN Compiler (v.2.8)

The SVS Definicon FORTRAN compiler (v.2.8) is produced by SVS of Cupertino, California. The linker is also made by SVS, and the loader is provided by the co-processor board manufacturer, Definicon Systems, Inc. The 32-bit FORTRAN compiler fully supports the Motorola MC 68020 Microprocessor, and the Motorola MC 68881 floating point co-processor. [Ref. 10:pp. E.1-E.3]

The SVS Definicon FORTRAN compiler is a two-step compiler where the source file (.FOR) is first compiled to an intermediate step before being compiled to an object file (.OBJ). This is in contrast to the SVS MS-DOS FORTRAN compiler which has two intermediate steps. The resulting object file is linked into an executable file by the linker. A special nuance of the Definicon system is that the loader must be used to move all of the 68020 executable file (compiler, linker, program) from the MS-DOS host computer to the Definicon DSI-780 board. The speed of the loader is directly related to the speed of the host system. Appendix C contains a list of all compiler switches and options. [Ref. 10:pp. E.1-E.3]

III. PROGRAM SOURCE CODE

A. THE FORTRAN PROGRAMMING LANGUAGE

The first American National Standard for the FORTRAN programming language was established in 1966 by the American National Standards Institute (ANSI) and was known as FORTRAN 66. However, most FORTRAN compilers that were based on this standard went by the name FORTRAN IV. The standard permitted the use of "compatible extensions," and by the mid 1970's it became apparent that many of the more commonly used extensions to FORTRAN 66 could be incorporated into a new FORTRAN language standard. In 1978, ANSI formally upgraded the FORTRAN language standard to what is commonly referred to as FORTRAN 77. Although the standard is voluntary, all modern FORTRAN compilers fully support the FORTRAN 77 standard. [Ref. 11:p. 427]

Most older FORTRAN programs, in particular NEC-3, were written before 1978. Because of this, it is important to understand the differences between FORTRAN 66 and FORTRAN 77, and how these differences will affect program portability.

1. Major Differences Between FORTRAN 77 and FORTRAN 66

a. Structured Branching Statements

The IF-THEN-ELSE-END IF control structure has been added to the language. Unlimited nesting of these control structures is allowed. Nesting within DO loops is also permitted. [Ref. 11:p. 480]

b. Character Data Type

A new data type, consisting of character strings of fixed declared length, has been added to the language, including character constants, character variables, and arrays of character data. Concatenation and designation of character substring is also supported. The Hollerith data type of ANSI x3.9-1966 has been deleted. [Ref. 11:p. 481]

c. DO Loop Changes

A DO statement specifying a terminal parameter whose value is less than that of the initial parameter is no longer prohibited, under such conditions the loop will be skipped and not executed. FORTRAN 66 specified that all loops must execute at least once. Negative increments are also permitted. Transfer of control into a DO loop is now prohibited. [Ref. 11:p. 481]

d. List Directed Input and Output

A form of input and output is provided which does not require an explicit format specification. [Ref. 11:p. 481]

e. Expressions

An arithmetic expression may include subexpressions of more than one type. The operands are first converted to the data type of the result. DO loop parameters can be any expression of integer, real, or double precision type. [Ref. 11:p. 481]

f. Compile-time Constants

The PARAMETER statement has been added, which declares the value corresponding to the symbolic name of a constant. Such a name may be used in an expression, DATA statement, or in following PARAMETER statements. [Ref. 11:p. 481]

g. IMPLICIT Type Declaration

An IMPLICIT statement may be used to modify the default typing by the compiler for variable and array names beginning with certain letters. [Ref. 11:p. 482]

h. Generic Intrinsic Functions

Most intrinsic functions such as SIN and COS now have a generic name, so that special names for complex and double precision (i.e., CSIN, DSIN) need not be used. [Ref. 11:p. 482]

i. Subprogram Reference

Subroutines and functions may contain ENTRY statements, and the alternate return may be used in subroutines. [Ref. 11:p. 482]

j. Array Bounds

Array declarations may include both upper and lower dimensions. Arrays may have up to seven dimensions. The last dimension of an array passed to a subroutine may be dimensioned inside the subroutine with an asterisk (*). FORTRAN 66 specified that the last array dimension of an assumed array be dimensioned with a one (1). (i.e., A(10,1) vs. A(10,*)) [Ref. 11:p. 482]

k. Computed GOTO Default

If the control expression of a computed GOTO is out of range, execution continues with the statement following the computed GOTO. [Ref. 11:p. 482]

l. INPUT/OUTPUT Statements

Several new features were added to the FORTRAN 77 standard to improve INPUT/OUTPUT. [Ref. 11:p. 482]

- An OUTPUT list may now contain constants and expressions.
- INPUT/OUTPUT statements may now contain a character string to be used as a format specification.
- END OF FILE and ERROR condition control.
- Tab format edit descriptors have been added.
- Direct access file support has been added.
- Character arrays maybe used as an internal file.
- OPEN, CLOSE and INQUIRE statements added.

m. SAVE Statement

The SAVE statement is now part of the standard. This statement instructs the compiler to retain the value of specified local variables between calls to the subprogram. [Ref. 11:p. 482]

n. FORTRAN Character Set/Comment Lines

The apostrophe (') and colon (:) are now part of the standard FORTRAN character set. Both the asterisk (*) and letter C in column 1 designate a comment line. [Ref. 11:p. 482]

2. Portability of FORTRAN Programs

FORTRAN is one of the few universal programming languages. For several decades it has been the primary computer language for scientific, numerical, and technical applications. Only now are some inroads into its position being made as simpler, better structured languages such as PASCAL, C, and ADA become more popular for numerical applications. FORTRAN's universality is based on the wide availability of compilers for virtually all sizes of computers from the super-computer to the microcomputer. While this has helped the exchange of FORTRAN programs, there have been portability problems. [Ref. 12:p. 134]

Many compiler manufacturers have introduced extensions to the language standard. These extensions fall into two broad categories, those which are true extensions of the power of the language such as new types of control structures and those which allow access to special hardware features of the system the compiler is supporting. Both types of extensions work against program portability, the ability to transfer programs from one computer to another with no source code modifications and to obtain the same results and level of precision. [Ref. 12:p. 134]

A second barrier to program portability is the FORTRAN standard itself, since not all parts of the language specifications are completely defined. For example, the depth to which DO loops can be nested is compiler-dependent. Such differences between compilers can lead to difficulties when moving programs to new computer systems. [Ref. 12:p. 135]

The best method of guaranteeing program portability is to strictly adhere to the standard when the program is created. This rarely occurs, however, since usually only one compiler is available when the program is written and the peculiarities of the compiler always seem to find their way into the program. There are six categories of the FORTRAN standard where most portability problems occur, language elements, expressions and assignments, control statements, specification statements, program units, and INPUT/OUTPUT. [Ref. 12:p. 135]

a. Language Elements

(1) Character Set. The standard defines upper case letters only, lower case should not be used except in comment lines. In addition, there are thirteen special characters:

blank , . ' : = + - * / ( ) $

and no other non-alphanumeric characters should be used. [Ref. 12:p. 136]

(2) Source Form. The standard defines a rigid source form, columns 1:5 for the label, column 6 for the continuation mark, columns 7:72 for the statement. This format should be respected. [Ref. 12:p. 136]

(3) Continuation Lines. The standard prohibits the presence of any non-blank characters in columns 1:5 of a continuation line; not all compilers enforce this, but the field should be left blank anyway. [Ref. 12:p. 136]

(4) Multiple Statement. Some compilers allow more than one statement to be written on a line, usually separated by the non-standard semi-colon (;). Not only is this non-portable, it is bad programming style. [Ref. 12:p. 137]

(5) In-line Comments. Some compilers allow comments to be added to statements on the same line following the non-standard exclamation mark (!). This is a UNIX style extension which should be avoided. [Ref. 12:p. 137]

(6) Long Names. Some compilers allow names to have more than six characters. While this is an improvement in style, it is also highly non-portable and very difficult to modify later. [Ref. 12:p. 137]

(7) Currency Symbol. The standard defines no particular role for the currency symbol ($). Some compilers, however allow it to be used as a character in names. This again is non-portable and should be avoided. [Ref. 12:p. 137]

(8) Logical Constants. The logical constants are .TRUE. and .FALSE.. Some compilers allow abbreviations such as .T. and .F. which should be avoided. [Ref. 12:p. 137]

(9) Character Strings. Character strings are delimited by apostrophes ('). No other character, in particular the non-standard double quotes ("), should ever be used. [Ref. 12:p. 138]

b. Expressions and Assignments

This section deals with some potential problems with expressions and assignments, mostly related to different hardware implementations of FORTRAN.

(1) Mixed Mode Relational Expressions. Although mixed mode relational expressions such as

```
IF (A/B .NE. I-J) THEN
```

are permitted by the standard, they should be avoided since real expressions will evaluate slightly differently on different machines. [Ref. 12:p. 138]

(2) Tests of Accuracy. It is also important to avoid tests of equality involving floating point operands, as hardware dependent rounding errors may cause them to be evaluated incorrectly. [Ref. 12:p. 138]

(3) Subscript Expressions. Some FORTRAN compilers allow the use of non-integer subscript expressions. The value of the expression is rounded according to the standard rules. Because rounding errors can cause different results on different machines, inherently real expressions should be rounded explicitly to the nearest integer value using the NINT intrinsic function, as in

```
A (NINT (P/Q)) = 0.
```

[Ref. 12:p. 139]

(4) Number of Subscripts. The number of subscripts in an array reference should be identical to the number of dimensions of the array as in

```
INTEGER   J(10,10)
:
J (3,4) = 0
```

and not:

```
J (33) = 0.
```

[Ref. 12:p. 139]

c. *Control Statements*

This section covers several areas where control statements can contribute to the non-portability of a program, and which should always be avoided.

(1) DO-loop Parameters. FORTRAN 77 permits the use of real and double precision quantities as DO-loop indices and parameters, both in evaluating the number of interactions and the successive values of the loop index. [Ref. 12:p. 140]

(2) Arrays in Nested Loops. Nested loops, when used, should be arranged so that the subscripted array's first element depends on the innermost loop and its last on the outermost loop, such as:

```
DO 1   J = 1, 10
    DO 2   I = 1, 10
                A (I,J) = B (I,J) + C (I,J)
    2  CONTINUE
1   CONTINUE
```

When loops are arranged in this manner, unexpected and dramatic losses in efficiency connected with the way words are stored in main memory can be avoided. [Ref. 12:p. 140]

(3) Loop Constructs. Many compilers now allow the use of PASCAL-type loop constructs, such as DO. . WHILE and DO. . UNTIL. These are non-portable and should never be used. [Ref. 12:p. 140]

(4) Branches into Control Constructs. Some compilers allow control to be passed into an IF-THEN block. This is bad style as well as being non-portable. [Ref. 12:p. 141]

d. Specification Statements

The section deals with the rules concerning specification statements.

(1) Data Types. The standard defined six data types, INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL, and CHARACTER. No other data types should be used, nor should non-standard declarations, such as REAL*4, or REAL*8 be used when a standard declaration has the same meaning. [Ref. 12:p. 141]

(2) EQUIVALENCE Statements. The EQUIVALENCE statement should never be used for variables of different types, since internal number representations differ from computer to computer, resulting in non-portable code. [Ref. 12:p. 141]

(3) PARAMETER Statements. Some compilers allow intrinsic function calls to be placed in constant expressions inside PARAMETER statements. This is not part of the standard and should be avoided. [Ref. 12:p. 141]

(4) DATA Statements. FORTRAN programs should never rely on the default data initialization of a specific machine/operating system combination. (for instance, the setting of all variables not explicitly defined by a DATA statement to zero). All variables should be explicitly set in either an assignment statement or DATA statement. [Ref. 12:p. 141]

e. Program Units

This section cover program units and the specification statements which are relevant to them.

(1) Main Program Header Line. All complete FORTRAN programs must contain a main program. The main program header line is optional, but if used it should not contain a parameter list. The parameter list was used in FORTRAN 66 as a means of defining the I/O files. All such files should be explicitly defined by the OPEN statement. [Ref. 12:p. 142]

(2) Intrinsic Functions. Most compilers offer additional intrinsic functions beyond those defined by the standard. These additional functions should never be used, since even if the same function were to be available on another system, the syntax would most likely be different. The most commonly available additional function is the TIME/DATE function. [Ref. 12:p. 142]

(3) Character-to-Integer Conversion. The values returned by the intrinsic functions CHAR and ICHAR are processor-dependent, and no portable program should depend on any specific values. [Ref. 12:p. 142]

(4) External Functions. Although not required to do so by the standard, all external functions should be explicitly declared in an EXTERNAL statement to avoid surprises. This is because the standard allows the compiler to assign any known function to an undeclared reference. [Ref. 12:p. 142]

(5) User-Defined Functions. User-defined functions should not alter the value of their arguments, or COMMON variables, nor perform I/O. The order in which any of these undesirable actions is performed is compiler dependent and leads to non-portable code. [Ref. 12:p. 123]

(6) COMMON Variables as Arguments. COMMON block variables should not be passed as arguments when the COMMON block is referenced in both the calling and called subroutine. It is especially hazardous to pass COMMON block variables when they are modified inside the called subprogram. [Ref. 12:p. 143]

(7) CHARACTER Variables in COMMON Blocks. The standard does not define any mapping of CHARACTER variables onto computer words as it does for other data types. Although permitted by many compilers, the mixing of CHARACTER variables with other data types in common blocks is highly non-portable since the size of a character storage unit varies from computer to computer. [Ref. 12:p. 143]

(8) Double-word Data Types in COMMON Blocks. Some compilers require that large data types, double precision and complex, should be positioned in memory so as to begin on an odd word. Since this is a grey area in the standard, all such variables should be positioned at the beginning of the common block so they will always be guaranteed of beginning on an odd word. [Ref. 12:p. 143]

(9) Names. Although allowed by some compilers, no COMMON block name should be the same as the name of any subprogram. [Ref. 12:p. 144]

(10) Order of Statements. While many compilers are lax in enforcing the order of statements as defined by the standard, all portable programs should strictly follow the correct order of statements. [Ref. 12:p. 144]

(11) SAVE Statement. The SAVE statement should be used as appropriate, as it might be required by some compilers on which the program will later run. [Ref. 12:p. 144]

(12) BLOCK DATA. When DATA statements are used for the initialization of COMMON variables, a BLOCK DATA subroutine should be used. [Ref. 12:p. 144]

f. Input/Output

This section deals with several points about I/O which can affect program portability.

(1) PRINT and READ statements. PRINT and READ statements without a unit specifier cause I/O to be directed to or from compiler-defined units, and, as such, they should not be used in portable code. WRITE and READ statements should use variables or symbolic constants for unit specifiers, since they can be easily modified if necessary. [Ref. 12:p. 144]

(2) Non-standard I/O. Many compilers have non-standard I/O statements which should never be used in portable programs. Non-standard parameters in standard I/O statements should also be avoided. [Ref. 12:p. 145]

(3) Parameter Lists. Both READ and WRITE statements have either positional or keyword parameters. Both styles should never be mixed in one statement. [Ref. 12:p. 145]

(4) I/O Unit Numbers. I/O unit numbers should never be explicit constants, or an asterisk (*), but rather symbolic constants or variables which can be easily changed when moving code from one system to another. [Ref. 12:p. 145]

(5) Format Specifications. Format Specifications may be stored in arrays. To avoid portability problems the arrays should always be of type CHARACTER. Additionally, format specifiers should always be separated by a comma (,) to ensure portability. [Ref. 12:p. 145]

(6) IOSTAT Parameter. The values returned by the IOSTAT parameters in I/O statements are compiler dependent and should never be explicitly enumerated; symbolic constants should be used instead. [Ref. 12:p. 145]

B. OVERVIEW OF THE NEC-3 FORTRAN PROGRAM

1. Theory

The NEC-3 is a highly capable, and versatile tool for the analysis of the electromagnetic response of antennas and other metal structures. The program computes directly the numerical solution of integral equations for currents induced on the structure by sources or incident fields, thus avoiding many of the simplifying assumptions required by other solution methods. [Ref. 1:p. 1]

The electric-field integral equation (EFIE) and the magnetic-field integral equation (MFIE) are both used to model the electromagnetic response of general

structures. Each equation has different advantages for a particular structure type. The EFIE is best suited for thin-wire structures of small or vanishing conductor volume. The MFIE, which does not work for thin-wire structures is most useful when modeling voluminous closed surface structures, particularly those with large smooth surfaces. The EFIE and MFIE are coupled for structures which contain both surfaces and wires. [Ref. 1:p. 3]

The integral equations are solved numerically using a form of the Method of Moments. The resulting current matrix equation is then solved by Gauss elimination and LU decomposition. The computation of the matrix elements and the solution of the matrix equation are by far the two most time-consuming steps in computing the response of a structure, usually accounting for 90% or more of the total computation time. [Ref. 1:p. 30-32]

The order of the matrix equation (N) is directly affected by the number of wire segments (N_s) and surface patches (N_p) in the structure being modeled. [Ref. 1:p. 30]

$$N = N_s + 2N_p \qquad \text{(III-1)}$$

When the structure is modeled near ground, penetrating the ground, or buried, the SOMNTX FORTRAN program is used to generate a data file containing the relevant Sommerfeld integral interpolation tables. By using a separate program to generate these interpolation tables,the computation time for ground related problems is greatly reduced. [Ref. 1:p. 37]

2. Program Organization

The NEC-3 program consists of over 10,000 lines of FORTRAN code, 89 subroutines and user-defined functions, and over 1000 variables. The program is entirely self-contained, i.e., no libraries are required. The SOMNTX program is somewhat more modest, being only 2000 lines long with only 25 subroutines and user-defined functions. Both programs were written largely in FORTRAN 66 and use archaic programming elements (GOTO, computed goto, assign, equivalence) extensively. Very little use is made of the advanced FORTRAN 77 control structures such as the IF-THEN-ELSE block. As a result, program flow is very difficult to follow.

C. MODIFICATIONS INCORPORATED INTO THE NEC-3 FORTRAN PROGRAM

The following are the modifications that have been incorporated into the NEC-3 and SOMNTX FORTRAN source code as part of this research. Most of the changes are related to the topics covered in the program portability section.

1. NEC-3 Modifications

a. SAVE Statements

Gerry J. Burke of the Lawrence Livermore Laboratory discovered that SAVE statements were required for all subprograms when he ported the VAX version to the MACINTOSH II. The justification for this is that in FORTRAN 66,

most compilers save all local variables automatically. By adding the SAVE statement to the NEC-3 the stochastic problem reported by Tim O'Hara disappeared.

b. Common Block Alignment

When the UNIX version of the MicroWay NDP FORTRAN compiler was used, it was discovered that a common block misalignment problem existed. The complex variable T1 was not defined as type COMPLEX in those subroutines where the variable was not referenced. However, since the common block GND was a part of those subroutines, it is mandatory that the variable T1 be explicitly declared as type COMPLEX in all subroutines where the common block GND is used.

c. Full Double Precision

The NEC-3 program was promoted to full double precision based on the recommendations of Tim O'Hara's research. This was done by placing the IMPLICIT REAL *8 (A-H, O-Z) statement at the beginning of all subprograms, and making the following intrinsic function replacements:

REAL	→	DBLE
FLOAT	→	DBLE
COMPLEX	→	COMPLEX*16
CMPLX	→	DCMPLX
AIMAG	→	DIMAG
CONJG	→	DCONJG

d. Generic Functions

As mentioned earlier, one of the improvements of FORTRAN 77 over FORTRAN 66 was the introduction of generic functions. To make the conversion of NEC-3 to double precision easier, all intrinsic functions except DBLE, DCMPLX, DIMAG, AND DCONJG were converted to their generic functions. For example, CABS became ABS.

e. FORTRAN 77 Standard Array Passing

FORTRAN 77 specifies that all assumed size dummy arrays use an asterisk, e.g., A(*), rather than the FORTRAN 66 method of dimensioning assumed size arrays with the number one (1), e.g., A(1). Since the old method was used extensively in NEC-3 and is not documented by any of the compilers used for this research, the author felt it prudent to convert all assumed size arrays to the FORTRAN 77 standard.

f. Output Statements

As covered previously, output statements without a unit specifier direct output to the default device, which is compiler-specific. In NEC-3, the PRINT statement sends the output to unit 6 when using the MicroWay compiler and to the screen when using the Lahey compiler. To ensure portability all PRINT statements were changed to WRITE statements.

g. External Functions

All external functions used in a program unit are now first declared in an EXTERNAL statement.

h. Subroutine Names

The names of two subroutines, INPUT and READ, conflicted with the names of routines in the SVS compiler's library. The solution was to rename the subroutines as INPUT1 and READ1, and also modify all the CALL statements.

*i. INTEGER*4, REAL*8, and COMPLEX*16*

All compilers make undeclared integer variables to be of type INTEGER*4, however, because all MS-DOS compilers now support the INTEGER*2 data type, the INTEGER statement is incorrectly interpreted by the compiler to be of type INTEGER*2 vice INTEGER*4. Severe problems occur whenever 4-byte integers are passed to 2-byte integers and vice versa. REAL*8 and COMPLEX*16 were used in lieu of DOUBLE PRECISION REAL and DOUBLE PRECISION COMPLEX, respectively, since all modern FORTRAN 77 compilers support these extensions.

j. Arrays of Mixed Data Types

In an effort to save space, the NEC-3 program reuses arrays whenever possible. Most of the time, real variables replace real variables, and no harm is done. In the DATA and DATAJ common blocks several integer arrays are

used to hold both integer and real variables. When single precision real variables are used, both real and integer variables are 4 bytes long, and can be used in the same array via an EQUIVALENCE statement without causing problems. However, this scheme does not work when double precision real variables are used.

In order to resolve conflicts, the REAL*8 arrays, T2X(1000), T2Y(1000), T2Z(1000), have been added to the COMMON block DATA. The following DIMENSION and EQUIVALENCE statements have been deleted:

- DIMENSION T2X(1000), T2Y(1000), T2Z(1000)
- EQUIVALENCE (T2X(1), ICON1(1)), (T2Y(1), ICON2(1)), (T2Z(1), ITAG(1))

The affected subroutines are: MAIN, ARC, CABC, CMNGF, CMSET, CMSS, CMSW, CMWS, CMWW, CONECT, DATAGN, ETMNS, FFLD, FFLDS, GFIL, GFLD, GFOUT, ISEGNO, LOAD, MOVE, NEFLD, NETWK, NFPAT, NHFLD, PATCH, QDSRC, RDPAT, REFLC, SBF, TBF, TRIO, and WIRE.

k. EQUIVALENCE Statements with Mixed Data Types

Like arrays, the mixing of data types of different lengths can cause severe problems. Two INTEGER variables in the DATAJ COMMON block were equivalenced to REAL*8 variables part of the time. The following changes were made:

- Add the REAL*8 variables T2YJ and T2ZJ to the DATAJ COMMON block.
- Delete the statement:
  EQUIVALENCE (T2YJ, IND1), (I2ZJ,IND2).

The affected subroutines are: CMNGF, CMSET, CMSS, CMSW, CMWS, CMWW, EFLD, ETMNS, HINTG, HSFLD, NEFLD, NHFLD, PCINT, QDSRC, SFLDS, UNERE.

l. File I/O Modifications

The SOMNTX data file is attached to unit 21. Arrays were written to the SOMNTX data file with the statement

WRITE (21, 100) A

where A is an array. The array is written correctly to the file, element by element, but when NEC-3 attempts to read in the array with the statement

READ (21, 100) A

a file read error results. The solution to the problem is to WRITE and READ all arrays with an implied DO LOOP. For example,

WRITE (21, 100) (A(I), I = 1, 10)

READ (21, 100) (A(I), I = 1, 10).

The affected subroutine is GNDINO.

m. Compiler-specific Subroutines

Two compiler-specific routines were added to NEC-3. The timer subroutine SECOND (ITIME), which returns the INTEGER*4 variable ITIME with the number of hundredths of seconds since midnight. The version subroutine VERSION prints a line of output which includes the compiler's name, date, and time of the output. Timer and version subroutines are listed in Appendix D.

2. SOMNTX Modifications

The following modifications were made to the SOMNTX program, the details are the same as for the NEC-3 program:

- SAVE statements added
- Generic functions
- FORTRAN 77 standard array passing
- Output statements
- External functions
- INTEGER*4
- File I/O

It should also be noted that the SOMNTX program was not promoted to full double precision for this research, as the accuracy of the output was sufficient without doing so. However, the output was extended to double precision prior to writing the data file so that the double precision NEC-3 could read the data file.

IV. RESULTS

The revised NEC-3 code was evaluated with three types of test problems. A set of simple problems, a mixed set of problems from the NEC user's guide, and a set of ground related problems. The NEC-3 input files for all problems tested are listed in Appendix E. Since the primary objective of this research was to evaluate the feasibility of moving a large mainframe-dependent scientific FORTRAN program to a 32-bit personal computer, four versions of NEC-3 (each compiler under a different compiler or math co-processor option) were tested for speed and accuracy against the IBM 3033AP mainframe computer.

In addition to the Definicon version, an attempt was made to compile the NEC-3 code with all three MS-DOS compilers using both math co-processors. Both of the SVS FORTRAN 386 compiler versions, however, repeatedly crashed while running the simple 49-segment dipole problem and were not further evaluated. In addition, the Lahey compiler does not support the DOUBLE PRECISION COMPLEX data type with the Weitek w3167 math co-processor.

A. DIPOLE EXAMPLES

Four dipole problems consisting of a 49-segment, a 99-segment, a 199-segment and a 299-segment dipole were run on all versions to evaluate the speed and

accuracy of each. Table 4.1 compares the full double precision version of NEC-3 with the single precision version. Tables 4.2-4.5 compare the fill time, factor time, and input impedance of each version against the mainframe. Table 4.6 compares the execution time of a 299-segment dipole with a 301-segment dipole.

TABLE 4.1

SINGLE VERSUS DOUBLE PRECISION
(LAHEY COMPILER)

EXAMPLE	FILL (SEC)	FACTOR (SEC)	INPUT IMPEDANCE
DIP 49			
SINGLE	5.66	.93	77.927 + j 43.850
DOUBLE	6.37	.99	77.900 + j 44.481
DIP 99			
SINGLE	21.70	7.47	79.041 + j 60.670
DOUBLE	24.27	8.41	77.997 + j 44.606
DIP 199			
SINGLE	83.93	60.20	77.906 + j 39.708
DOUBLE	94.14	67.45	78.055 + j 44.687
DIP 299			
SINGLE	186.80	203.50	85.347 + j 12.047
DOUBLE	209.38	228.49	78.077 + j 44.721
TOTAL TIME			
SINGLE	570.19		
DOUBLE	639.5		
DOUBLE PRECISION TIME PENALTY → ≈ 12%			

TABLE 4.2

49-SEGMENT DIPOLE

SYSTEM	FILL TIME	FACTOR TIME	INPUT IMPEDANCE
IBM 3033 AP	3.01	0.24	77.90 + j 44.48
AST 386/33			
LAHEY w/386	6.37	0.99	77.90 + j 44.48
NDP w/387 w/3167	7.86 5.93	0.77 0.44	77.90 + j 44.48 77.90 + j 44.48
DIS-780	30.43	4.67	77.90 + j 44.48

TABLE 4.3

99 SEGMENT-DIPOLE

SYSTEM	FILL TIME	FACTOR TIME	INPUT IMPEDANCE
IBM 3033 AP	12.25	2.24	78.00 + j 44.61
AST 386/33			
LAHEY w/386	24.27	8.41	78.00 + j 44.61
NDP w/387 w/3167	29.93 22.46	6.27 3.9	78.00 + j 44.61 78.00 + j 44.61
DIS-780	117.33	38.22	78.00 + j 44.61

TABLE 4.4

199-SEGMENT DIPOLE

SYSTEM	FILL TIME	FACTOR TIME	INPUT IMPEDANCE
IBM 3033 AP	51.82	19.00	78.05 + j 44.69
AST 386/33			
LAHEY w/386	94.14	67.45	78.05 + j 44.69
NDP w/387 w/3167	115.07 85.85	51.30 32.08	78.05 + j 44.69 78.05 + j 44.69
DIS-780	456.33	308.18	78.05 + j 44.69

TABLE 4.5

299-SEGMENT DIPOLE

SYSTEM	FILL TIME	FACTOR TIME	INPUT IMPEDANCE
IBM 3033 AP	121.69	67.36	78.08 + j 44.72
AST 386/33			
LAHEY w/386	209.38	228.49	78.08 + j 44.72
NDP w/387 w/3167	255.35 190.04	175.93 109.52	78.08 + j 44.72 78.08 + j 44.72
DIS-780	1017.99	1067.64	78.08 + j 44.72

TABLE 4.6

EXECUTION TIMES OF 299-SEGMENT VS. 301-SEGMENT DIPOLES
(Lahey Compiler)

PROBLEM	FILL TIME	FACTOR TIME	INPUT IMPEDANCE (Ω)
299	209.38	228.49	78.08 + j 44.72
301	256.88	418.09	78.08 + j 44.72

As demonstrated in Table 4.1, the full double precision version of NEC-3 is only 12% slower than the single precision version. However, a 12% decrease in speed is small when compared to the significant improvement in accuracy. Tables 4.2 through 4.5 show that all double precision versions of the code are equally accurate. The two versions using the Intel 80387 math co-processor, Lahey and MicroWay, differ in execution speed by less than 1%. The Microway Weitek version is 30% faster than the MicroWay Intel 80387 version, with no loss in accuracy. The Definicon system is 4.7 times slower than the Intel 80387 versions.

Table 4.6 illustrated how execution times increase dramatically when the number of segments in the problem exceeds the in-core memory limit of 300 segments. This increase in execution time is directly a function of hard disk speed and data transfer rates. While microcomputer hard disk systems have improved

dramatically over the past few years, they are still significantly slower than the large hard disk systems used by mainframe computers. Because of this, large problems requiring an out-of-core solution are best performed on a mainframe computer.

Although very slow in comparison to the AST 386/33, the Definicon DSI-780 co-processor board proves that a full double precision 32-bit version of NEC-3 with 300 segments in-core can be executed in a mere 4 megabytes of RAM. Since the Definicon board resides in a host system, the operating system overhead is not included in this total. For comparison, assuming the operating system requires a full 1 megabyte of RAM, the double precision version of NEC-3 with 300 segment in-core limit uses only 5 megabytes of RAM on the AST 386/33. If the in-core limit was raised to 500 segments, only 7.5 megabytes of RAM would be required, 750 segments: 12.4 megabytes of RAM, and all 1000 segments would require only 20 megabytes of RAM. While 20 megabytes of RAM costs a great deal of money, this cost is less than half the original cost of the AST computer system.

B. BASIC ABOVE GROUND PROBLEMS

The following examples were all taken from the NEC user's guide, Reference 13, with the exception of the first example.

1. Rhombic Antenna

This example is a horizontally polarized antenna modeled over a finite ground. This significant output data is tabulated in Table 4.7.

TABLE 4.7

RHOMBIC ANTENNA HORIZONTALLY POLARIZED

TAG NUMBER	SEGMENT NUMBER	INPUT IMPEDANCE Ω	
		REAL	IMAGINARY
1	1	352.05	172.06
3	81	352.05	172.06
HORIZONTAL GAIN NORMALIZATION FACTOR = 17.95 db			

All versions tested produced exactly the same results. The execution times for each problem varied the same between each version as the dipole problems did, thus the fill and factor times are not included in the table summaries.

2. NEC User's Guide Problems

The significant results from the first nine problems in the NEC user's guide, Reference 13, are presented here in Tables 4.8 through 4.16. With the

exception of problems 3, and 5 the output of all versions of NEC-3 exactly match the official validated results listed in the NEC user's guide.

TABLE 4.8

CENTER FED LINEAR ANTENNA
(Example 1 in Ref. 13)

LOAD (Ω)	INPUT IMPEDANCE (Ω)	
	REAL	IMAGINARY
UNLOADED	82.698	+ j 46.306
100	92.698	+ j 41.941

SEGMENT NUMBER	CURRENT (A)		CHARGE DENSITY (C/m)	
	MAG	PHASE	MAG	PHASE
1	2.702E-03	-30.68	3.665E-11	60.06
4	9.829E-03	-24.34	5.691E-19	24.74

NEAR ELECTRIC FIELDS

LOCATION Z.2(m)	EX		EZ	
	MAG (V/m)	PHASE	MAG (V/m)	PHASE
0.0	1.023E-05	24.74	1.304E+01	-175.10
0.0179	5.544E+01	-66.31	1.254E+01	-175.08
0.0714	2.127E+02	-100.30	4.214E-04	-6.13
0.1429	4.384E+02	-113.51	3.091E-04	-94.14
0.2500	5.520E+02	-121.29	3.803E+02	-121.43

TABLE 4.9

CENTER FED LINEAR ANTENNA, VARIABLE FREQUENCY
(Example 2 in Ref. 13)

FREQUENCY (MHz)	UNNORMALIZED IMPEDANCE (Ω)	NORMALIZED IMPEDANCE (Ω)
200	2.658E+01 - j 6.321E+02	5.315E-01 - j 1.264E+01
250	4.714E+01 - j 2.724E+02	9.429E-01 - j 5.447E+00
300	8.055E+01 + j 4.571E+01	1.611E+00 + j 9.143E-01
FREQUENCY (MHz)	**CONDUCTIVITY (MHOS/m)**	**INPUT IMPEDANCE (Ω)**
300	3.72E+07	1.124E+02 + j 6.543E+01

TABLE 4.10

VERTICAL HALF-WAVE DIPOLE OVER GROUND
(Example 3 in Ref. 13)

GROUND TYPE	INPUT IMPEDANCE (Ω)	VERTICAL GAIN NORMALIZATION FACTOR (db)
PERFECT	1.064E+02 + j 9.905E+0	8.52
FINITE	1.111E+02 + j 1.101E+01	1.54

NORMALIZED RECEIVING PATTERN
NORMALIZATION FACTOR = 0.3277E-01

THETA	PATTERN (db)
0	-999.99
10	-14.42
20	-8.73
30	-6.02
40	-5.00
50	-4.44
60	-2.33
70	0.00
80	-0.90
90	-999.99

NOTE = The radiated field near ground for this example differed from that listed in Reference 13 by 5% to 10%.

TABLE 4.11

T-ANTENNA ON A BOX OVER PERFECT GROUND
(Example 4 in Ref. 13)

INPUT IMPEDANCE (Ω)	AVERAGE POWER GAIN (db)
1.807E+02 + j 2.177E+02	1.7999

TABLE 4.12

12 ELEMENT LOG PERIODIC ANTENNA IN FREE SPACE
(Example 5 in Ref. 13)

INPUT IMPEDANCE (Ω)	NORMALIZED GAIN FACTOR (db)
4.246E-01 - j 7.984E-01	9.76
NOTE: Imaginary portion of input impedance is incorrect, the correct answer is - j 4.484E-01. Radiation patterns differ from that of Reference 13 by 5% to 10%.	

TABLE 4.13

CYLINDER WITH ATTACHED WIRE
(Example 6 in Ref. 13)

TAG NUMBER	SEGMENT NUMBER	INPUT IMPEDANCE (Ω)
1	1	1.799E+01 - j 1.179E+02
2	5	5.493E+01 + j 2.527E+01

ISOLATION DATA

COUPLING BETWEEN

TAG NUMBER	SEGMENT NUMBER	TAG NUMBER	SEGMENT NUMBER
1	1	2	5

MAXIMUM COUPLING (db)	FOR MAXIMUM COUPLING	
	LOAD IMPEDANCE (Ω)	INPUT IMPEDANCE (Ω)
-13.709	55.83 - j 20.64	18.23 - j 11.64

TABLE 4.14

SCATTERING BY A WIRE
(Example 7 in Ref. 13)

RADIATION PATTERNS			
FREE SPACE, EXCITATION 0°			
ANGLE THETA	POWER GAIN		
	VERT (db)	HORZ (db)	TOTAL (db)
0	-12.48	-999.99	-12.48
45	-18.33	-999.99	-18.33
FREE SPACE, EXCITATION 45°			
ANGLE THETA	POWER GAIN		
	VERT (db)	HORZ (db)	TOTAL (db)
0	-18.38	-999.99	-18.38
45	-10.40	-999.99	-10.40
PERFECT GROUND, EXCITATION 45°			
ANGLE THETA	POWER GAIN		
	VERT (db)	HORZ (db)	TOTAL (db)
80	-36.78	-999.99	-36.78
-20	-8.98	-999.99	-8.98
-30	-8.95	-999.99	-8.95
-80	-37.90	-999.99	-37.90
FINITE GROUND, EXCITATION 45°			
ANGLE THETA	POWER GAIN		
	VERT (db)	HORZ (db)	TOTAL (db)
80	-21.13	-999.99	-21.13
0	-19.82	-999.99	-19.92
-80	-22.11	-999.99	-22.11

TABLE 4.15

STICK MODEL OF AN AIRCRAFT IN FREE SPACE
(Example 8 in Ref. 13)

EXCITATION	RADIATION PATTERNS			
THETA	ANGLE THETA	POWER GAINS		
		VERT (db)	HORZ (db)	TOTAL (db)
0	0	-2.98	-999.99	-2.98
90	90	-51.77	-9.79	-9.79

TABLE 4.16

BISTATIC SCATTERING BY A SPHERE
(Example 9 in Ref. 13)

RADIATION PATTERNS				
ANGLES		POWER GAINS		
THETA	PHI	VERT (db)	HORZ (db)	TOTAL (db)
90	0	-4.45	-999.99	-4.45
0	0	-8.67	-999.99	-8.67
-90	0	7.85	-999.99	7.85
NEAR ELECTRIC FIELDS				
LOCATION			EZ	
X(m)	Y(m)	Z(m)	MAG (V/m)	PHASE
0	0	0	0.9962	0.03
0	20	0	0.9953	-0.02
0	25	0	0.9955	-0.05
0	50	0	0.9825	9.30

While the errors in running problems 3 and 5 from the NEC user's guide are significant, they are not catastrophic. In fact, the calculated answers differ from the correct answers by only 5% to 10%. What is also unusual about the problem is that all versions of the double precision NEC-3 generated exactly the same answers, including the incorrect ones. Furthermore, when the mainframe NEC-3 code was recompiled with the new IBM VS2 FORTRAN compiler, the results were exactly the same as the personal computer versions, including the inaccuracies of Problems 3, and 5. This demonstrates that the problem is not hardware or compiler related, but rather related to how NEC-3 was written. It is possible the new IBM (VS2) and personal computer FORTRAN 77 compilers somehow implement data structures such as COMMON blocks and EQUIVALENCE statements in a slightly different manner than the old IBM (VS1) compiler. It is also possible that the problem is a result of a non-rigorously written section of code. Since much of the NEC-3 program was written in the style of FORTRAN 66, the less tolerant FORTRAN 77 compilers might have incorrectly compiled a small section of the NEC-3 code.

C. GROUND PROBLEMS

Tables 4.17 through 4.21 tabulate the significant results of several ground related problems. As with the previous example, all versions of NEC-3 produced exactly the same correct results.

TABLE 4.17

MONOPOLE OVER A RADIAL WIRE SCREEN OVER FINITE GROUND

SCREEN HEIGHT (m)	INPUT IMPEDANCE (Ω)	AVERAGE POWER GAIN
0.2	83.05 - j 3346.5	0.5468
0.6	224.1 - j 86.80	0.3876

TABLE 4.18

DIPOLE ANTENNA MOVING CLOSER TO GROUND

HEIGHT ABOVE GROUND (m)	INPUT IMPEDANCE (Ω)
0.1	66.34 + j 62.50
0.03	73.89 + j 75.49
0.01	97.87 + j 127.95
0.003	119.87 + j 219.62
0.001	132.87 + j 322.13
0.0003	144.21 + j 441.81
0.0001	154.59 + j 554.56
0.00003	167.34 + j 681.82
0.00001	180.94 + j 802.05
3.0E-6	198.95 + j 939.39
1.0E-6	219.38 + j 1071.00
3.0E-7	249.11 + j 1223.50 ***
1.0E-7	280.04 + j 1361.90 ***
*** NOTE: Integration was step size limited, although impedance calculations are correct.	

TABLE 4.19

DIPOLE IN DIELECTRIC AND LOSSY MEDIA

CONDUCTIVITY = 0.0 INPUT IMPEDANCE = 11.50 - j 32.66			
RADIATION PATTERNS			
ANGLE THETA	POWER GAINS		
	VERT (db)	HORZ (db)	TOTAL (db)
10	-14.53	-999.99	-14.53
50	-0.88	-999.99	-0.88
90	1.99	-999.99	1.99
CONDUCTIVITY = 0.89E-02 **INPUT IMPEDANCE = 70.34 + j 11.02**			
RADIATION PATTERNS			
ANGLE THETA	POWER GAINS		
	VERT (db)	HORZ (db)	TOTAL (db)
10	-21.56	-999.99	-21.56
50	-8.07	-999.99	-8.07
90	-5.30	-999.99	-5.30

TABLE 4.20

MONOPOLE ON A GROUND STAKE

INPUT IMPEDANCE = 96.74 +j 38.41 (Ω) AVERAGE POWER GAIN = 0.3215 db				
NEAR ELECTRIC FIELDS				
LOCATION Z(m)	EX		EZ	
	MAG (V/m)	PHASE	MAG (V/m)	PHASE
0	0.172E-05	-8.43	0.106E-04	-45.05
90	0.121E-05	-13.27	0.112E-04	33.01
100	0.116E-05	-16.06	0.121E-04	36.96
200	0.103E-05	-70.9	0.223E-04	43.20

TABLE 4.21

MONOPOLE ON A 6-WIRE RADIAL GROUND SCREEN

INPUT IMPEDANCE = 53.96 + j 29.45 (Ω) AVERAGE POWER GAIN = 0.5821 (db)				
NEAR ELECTRIC FIELDS				
LOCATION Z(m)	EX		EZ	
	MAG (V/m)	PHASE	MAG (V/m)	PHASE
0	0.293E-05	-14.51	0.179E-04	-51.14
10	0.282E-05	-13.51	0.163E-04	-42.66
130	0.178E-05	-33.65	0.258E-04	37.66
140	0.174E-05	-38.53	0.276E-04	38.68
200	0.175E-05	-76.98	0.380E-04	37.11

V. CONCLUSIONS

The results of testing the double precision version of NEC-3 with a wide variety of problems, clearly show that NEC-3 can be successfully implemented on a 32-bit personal computer.

The AST Premium 386/33 utilizing the Intel 80387 math co-processor is 2.3 times slower than the IBM 3033 mainframe. When using the Weitek w3167, the AST Premium 386/33 is only 1.62 times slower than the mainframe. Since the fill and factor times of the IBM 3033AP mainframe are true processor times, without the obvious time-sharing overhead added to them, the speed of the AST Premium 386/33 is very close to the practical speed of a moderately loaded mainframe computer.

The full double precision NEC-3 code exactly matched the output of the IBM 3033AP mainframe for accuracy. The additional time required to perform all calculations in double precision arithmetic was shown to be only 12% more than the full single precision version.

The Definicon DSI-780 demonstrated that the full double precision NEC-3 can be successfully executed in 4 MB of RAM. When the operating system is taken into account, the full double precision NEC-3, with 300 unknowns in-core, can

be operated on any 32-bit personal computer with at least 5 MB of RAM. If the in-core limit were raised from the current 300 unknown to 1000 unknowns, the double precision NEC-3 code could be operated on a system with at least 20 MB of RAM. Due to the inherent slow speed of microcomputer hard disk systems relative to the mainframe, it is recommended that the NEC-3 code be converted to the largest possible number of unknowns in-core for use on personal computers.

The inaccuracies discovered while running the sample problems from the NEC User's Guide indicate that the "bug" is not hardware or software dependent, but a matter of how the new FORTRAN compilers are implemented. It is possible that the solution to the problem may lie in how the new FORTRAN compilers implement shared memory, since NEC-3 uses shared arrays via COMMON blocks and EQUIVALENCE statements throughout the program. It is recommended that the code be rewritten to eliminate the use of EQUIVALENCE statements for reusing arrays.

APPENDIX A

LAHEY F77L3 FORTRAN COMPILER

Switches and Options

/O	IMPLICIT NONE in effect
/NO	Standard implicit typing in effect
/7	Load NDP (80287 or 80387) nonINTEGER operands from memory
/N7	NDP nonINTEGER operands may be kept in the NDP between statements
/B	Bounds check array subscripts
/NB	No bounds checking
/C	Output warning messages indicating statements that do not conform to the FORTRAN 77 Standard
/NC	Suppress the output of warning messages indicating nonconformances to the FORTRAN 77 Standard
/D	Direct files are processed without headers
/ND	Direct files require F77L-EM/32-generated headers
/F	Free-format source file
/NF	Standard-format source file
/H	Hardcopy (source listing) output
/NH	No hardcopy (listing) output
/I	Subprogram interface check
/NI	No subprogram interface check
/K[H]	Generate Weitek 1167 and 3167 code
/NK	No 1167 and 3167 code generated
/L	Line-number table
/NL	No line-number table
/O	Output compiler options
/NO	No options output
/P	Protect constant arguments
/NP	No protection
/Q1	Ensure code never uses more that 8 NDP registers at a time
/NQ1	Infinite NDP stack for expressions
/Q2	Create the protected-mode side of an RPC interface module

/NQ2	Don't create the protected-mode side of an RPC interface module
/Q3	Create the real-mode side of an RPC interface module
/NQ3	Don't create the real-mode side of an RPC interface module
/R	Remember (SAVE) subprogram local variables and arrays.
/NR	No remembering local items
/S	Generate SOLD3 information file
/NS	No SOLD3 information file generated
/T	Type; default length of INTEGER type is 2, LOGICAL type is 1
/NT	Default length of INTEGER is 4, LOGICAL is 4
/W	Warning messages output
/NW	Not all warning messages output
/X	Cross-reference listing
/NX	No cross-reference listing
/Z1	Perform production code optimizations (limits SOLD3 and P77L Profiler interfaces)
/NZ1	No optimizations that would limit SOLD3 and P77L Profiler interfaces

APPENDIX B

MICROWAY NDP FORTRAN-386 COMPILER

Switches and Options

-c	Compile to the "obj" level only
-list	Create a ".lst" file with errors annotated
-LIST	Same as above, but listing does not show pathnames for source
-Idir	Alternate directory(ies) for INCLUDE files
-o filename	Place executable file output into the file named: "filename"
-R	Put all data in the code segment, i.e., generate "ROMable" code
-S	Do not produce object files or executable files
-v	Have the compiler driver print out the program name and command-line arguments as it runs each subprocess
-w	Suppress warning messages
-lname	Search the library named c:\NDP\libname.lib
-symbols	Place the symbols command into the linker file for Phar Lap linker
-g	Turn on trace/debug information
-ga	Generate a frame pointer for stack traces
-rt1	Allow arbitrary file names to be specified to the compiler
-rt2	Display the names of files as they are opened
-rt3	Do not stop in the event of a code generator abort
-rt4	Do not use 386 inline library routines
-n0	80287 inline code with library calls for transcendentals
-n1	80287 inline code with inline transcendentals (except sin and cos)
-n2	80387 inline code with all transcendentals (including sin and cos)
-n3	Advanced 80387 stack utilization
-n4	Weitek W1167 code
-n5	Produce W1167 macro instructions

-n6	Promote no-Float switch for Weitek
-CG1	Turn on runtime checking of subranges and array bounds
-CG2	Allocate all variable to memory
-CG3	Allocate code temporaries to memory
-CG4	Prepend all variables with an underscore
-CG5	Output an assembly file using ".asm" extension; the default is ".s"
-CG6	Do not put an underscore in front of the names of global variables and procedures
-CG7	Avoid jumps with inline code
-CG8	Turn of jump shortening
-CG9	Lengthen bytes in jump
-CG10	Suppress version number in data segment
-O	Perform default optimizations
-OL	Add speed optimizations related to moving code out of loops and speeding up loops in general
-OM or -OLM	Add additional memory optimizations to -O and -OL
-ONW	Emit a warning when dead code is eliminated
-OFF	Activates -OFFP, -OFFR, -OFFA, -OFFH, -OFFS and -OFFN
-OFFP	Disable peephole optimizer
-OFFR	Do not allocate programmer-defined local variable to a register unless they are declared register
-OFFA	Do not move frequently used procedure and data addresses into registers
-OFFH	Turn off the code-hoisting optimization
-OFFS	Turn off the optimization that deletes all code that stores into or modifies variables that are never read from
-OFFN	Do not move invariant floating-point expressions out of loops
-i2	Make the type INTEGER be INTEGER*2
-onetrip	Execute at least one iteration of every DO loop
-u	Make "undefined' the default data type for undeclared variables
-ep	Allow inline comments with "!" as in VAX FORTRAN
-U	Do not convert upper-case names in FORTRAN to lower-case
-f1	This makes characters unsigned as they are in some implementations of FORTRAN

-f2	Turn of compiler-time checking of FORMAT statements
-f3	Pad Hollerith constants on the right with blanks
-f4	Compile lines starting with "x", "X", "d", "D".
-f5	Do not allow dollar signs in names

APPENDIX C

SVS FORTRAN COMPILERS

Switches and Options

+q -q	Show more (-q) or less (+q) information on the progress of the compile to the user
+p -p	Prompt (+p) or don't prompt (-p) to the standard input in the case of a compile time error
+x	Generate a cross reference in the listing file
+d -d	Controls whether the compiler does (+d) or does not (-d) generate tables of information for the SVS Debugger
+c72	Truncate input lines to 72 columns
+save	Compile program as if each subprogram contained a SAVE statement
+saveall	Same as setting the $SAVEALL option
+charequ	Allow certain non ANSI Standard associations between character and numeric data
+int2	Change default attribute of INTEGERs and LOGICALs
+log2	Change default length attribute of LOGICALs
+dc -dc	Conditionally compile debugging source code found in lines with the letter D in column 1.
+f -f	Generate code for floating point hardware (+f) or floating point software (-f)
+287	Generate Intel 287 floating point code, on Intel 386 only
+387	Generate Intel 387 floating point code, on Intel 386 only
+w1167	General Weitek 1167 coprocessor floating point code, on Intel 386 only
-l*f*name	Create a listing file of the source program in the file named *f*name
-e*f*name	Place a summary of the compile time errors on file named *f*name
-i*f*name	Name the ".i" file *f*name

+b -b	Control interpretation of the 'UNFORMATTED' file FORM in OPEN statements
+e -e	The +e and -e option control whether the arguments to subroutines and functions are popped in the exit code of the routine before return (+e) or by the caller after return from the routine (-e)
+cc -cc	Control of parameter passing order and poping of parameters in subroutine calls
+a -a	Flag non-ANSI Standard features which are utilized by the program
+c -c	Setting +c instructs the compiler to create an information file of the source file positions of compile time errors

APPENDIX D

TIMER AND VERSION SUBROUTINES

1. LAHEY F77L3-EM/32

```
      INCLUDE C:\NDP\DNEC3.F

      SUBROUTINE VERSION
      CHARACTER*11 CTIME
      CHARACTER*8 CDATE
      CALL DATE(CDATE)
      CALL TIME(CTIME)
      WRITE (6,100) CDATE,CTIME
100   FORMAT (//,36X,'VERSION - LAHEY DOUBLE W/SAVES   DATE:',A,3X
     &,'TIME:',A,//)
      END

      SUBROUTINE SECOND(TIME)
      INTEGER*4 TIME
      CALL TIMER(TIME)
      RETURN
      END
```

2. MICROWAY NDP FORTRAN386

```
      INCLUDE "DNEC3.F"

      SUBROUTINE VERSION
      CHARACTER*11 CTIME
      CHARACTER*8 CDATE
      CALL DATE(CDATE)
      CALL TIME(CTIME)
      WRITE (6,100) CDATE,CTIME
100   FORMAT (//,36X,'VERSION - MICROWAY DOUBLE W/SAVES   DATE:',A,3X
     &,'TIME:',A,//)
      END

      SUBROUTINE SECOND(ITIME)
      INTEGER*4 ITIME,YEAR,MONTH,DAY,HOUR,MINUTE,SECOND,MICROSEC
      CALL TIMEDATE(YEAR,MONTH,DAY,HOUR,MINUTE,SECOND,MICROSEC)
      ITIME = HOUR*360000 + MINUTE*6000 + SECOND*100 + MICROSEC*1E-4
      RETURN
      END
```

3. SVS FORTRAN386

```
$INCLUDE C:\NDP\DNEC3.F

      SUBROUTINE VERSION
      WRITE (6,100)
100   FORMAT (//,36X,'VERSION - SVS (MSDOS) W/SAVES ',//)
      END

      SUBROUTINE SECOND(ITIME)
      INTEGER*4 ITIME,IHR,IMIN,ISEC,I100
      CALL GETTIM(IHR,IMIN,ISEC,I100)
      ITIME = IHR*360000 + IMIN*6000 + ISEC*100 + I100
      RETURN
      END
```

4. SVS DEFINICON FORTRAN

```
$SYSTEM
$INCLUDE C:\DNEC3.F

      SUBROUTINE SECOND(ITIME)
      INTEGER*4 ITIME,HOUR,MIN,SEC,HUND,TIME

      TIME=_STIME()
      HOUR = ISHFT(TIME,-24)
      MIN = ISHFT(IAND(16711680,TIME),-16)
      SEC = ISHFT(IAND(65280,TIME),-8)
      HUND = IAND(255,TIME)
      ITIME = HOUR*360000 + MIN*6000 + SEC*100 + HUND
      RETURN
      END

      SUBROUTINE VERSION
      WRITE (6,100)
100   FORMAT (//,36X,'VERSION - SVS-DEFINICON DOUBLE W/SAVEALL ',//)
      END
```

APPENDIX E

NEC TEST PROBLEMS

```
CE    DIPOLE WITH 49 SEGMENTS
GW 1,49,0,0,0,0,0,.5,.00001
GP
GE
EX 0,1,25
PT -1
XQ
NX
CE    DIPOLE WITH 99 SEGMENTS
GW 1,99,0,0,0,0,0,.5,.00001
GP
GE
EX 0,1,50
PT -1
XQ
NX
CE    DIPOLE WITH 199 SEGMENTS
GW 1,199,0,0,0,0,0,.5,.00001
GP
GE
EX 0,1,100
PT -1
XQ
NX
CE    DIPOLE WITH 299 SEGMENTS
GW 1,299,0,0,0,0,0,.5,.00001
GP
GE
EX 0,1,150
PT -1
XQ
EN
```

```
CM RHOMBIC ANTENNA HORIZONTALLY POLARIZED
CM LEG LENGTH=398.0 FT.
CM CENTER WIDTH=314.0 FT.
CM APEX ANGLE=44.0 DEGREES.
CM HEIGHT ABOVE GROUND=160.0 FT.
CM GROUND PARAMETERS-EPSILON=80. SIGMA=4. MHOS/M. (SEA WATER)
CE CONDUCTOR-AWG NO. 10 WIRE DIA.=0.00425 FT.
GW1,40,0.0,0.0,160.0,366.082,157.0,160.0,0.00425
GW2,40,366.082,157.0,160.0,732.164,0.0,160.0,0.00425
GX2,010
GS0,0,0.304801
GE1
FR0,0,0,0,10.0
GN0,0,0,0,80.0,4.0
LD0,2,40,40,300.0
LD0,4,40,40,300.0
EX0,1,1,0,0.5
EX0,3,1,10,-0.5
RP0,37,1,1401,90.0,0.0,-5.0,0.0
EN
```

```
CEEXAMPLE 1.  CENTER FED LINEAR ANTENNA.
GW0, 7, 0., 0., -.25, 0., 0., .25, .001
GE
EX0, 0, 4, 0, 1.
XQ
LD0, 0, 4, 4, 10., 3.E-09, 5.3E-11
PQ
NE0, 1, 1, 15, .001, 0., 0., 0., 0., .01786
NX
CMEXAMPLE 2. CENTER FED LINEAR ANTENNA.
CM           CURRENT SLOPE DISCONTINUITY SOURCE.
CM           1. THIN PERFECTLY CONDUCTING WIRE
CE           2. THIN ALUMINUM WIRE
GW0,8,0.,0.,-.25,0.,0.,.25,.00001
GE
FR0,3,0,0,200.,50.
EX5,0,5,1,1.,0.,50.
XQ
LD5,0,0,0,3.720E+07
FR0,1,0,0,300.
EX5,0,5,0,1.
XQ
NX
CMEXAMPLE 3.  VERTICAL HALF WAVELENGTH OVER GROUND
CM            EXTENDED THIN WIRE KERNEL USED
CM            1. PERFECT GROUND.
CM            2. IMPERFECT GROUND INCLUDING GROUND WAVE AND RECEIVING
CE               PATTERN CALCULATIONS
GW0,9,0.,0.,2.,0.,0.,7...3
GE1
EK
FR0,1,0,0,30.
EX0,0,5,0,1.
GN1
RP0,10,2.1301,0.,0.,10.,90.
GN0,0,0,0,6.,1.000E-03
RP0,10,2,1301,0.,0.,10.,90.
RP1,10,1,0,1.,0.,2.,0.,10.000E+04
EX1,10,1,0,0.,0.,0.,10.
PT2,0,5,5
XQ
EN
```

```
CEEXAMPLE 4.  T ANTENNA ON A BOX OVER PERFECT GROUND
SP0,0,.1,.05,.05,0.,0.,.01
SP0,0,.05,.1,.05,0.,90.,.01
GX0,110
SP0,0,0.,0.,.1,90.,0.,.04,.0
GW1,4,0.,0.,.1,0.,0.,.3,.001
GW2,2,0.,0.,.3,.15,0.,.3,.001
GW3,2,0.,0.,.3,-.15,0.,.3,.001
GE1
GN1
EX0,1,1,0,1.
RP0,10,4,1001,0.,0.,10.,30.
NX
CM  EXAMPLE 5.
CM 12 ELEMENT LOG PERIODIC ANTENNA IN FREE SPACE.
CM   78 SEGMENTS.     SIGMA=D/L       RECEIVING AND TRANS. PATTERNS.
CM DIPOLE LENGTH TO DIAMETER RATIO=150.
CE TAU=0.93.    SIGMA=0.70.     BOOM IMPEDANCE=50. OHMS.
GW 1,5,0.,-1.,0.,0.,1.,0.,0.00667
GW 2,5,-.7527,-1.0753,0.,-.7527,1.0753,0.,.00717
GW 3,5,-1.562,-1.1562,0.,-1.562,1.1562,0.,.00771
GW 4,5,-2.4323,-1.2432,0.,-2.4323,1.2432,0.,.00829
GW 5,5,-3.368,-1.3368,0.,-3.368,1.33368,0.,.00891
GW 6,7,-4.3742,-1.4374,0.,-4.3742,1.4374,0.,.00958
GW 7,7,-5.4562,-1.5456,0.,-5.4562,1.5456,0.,.0103
GW 8,7,-6.6195,-1.6619,0.,-6.6195,1.6619,0.,.01108
GW 9, 7,-7.8705,-1.787,0.,-7.8705,1.787,0.,.01191
GW10,7,-9.2156,-1.9215,0.,-9.2156,1.9215,0.,.01281
GW11,9,-10.6619,-2.0662,0.,-10.6619,2.0662,0.,.01377
GW12,9,-12.2171,-2.2217,0.,-12.2171,2.2217,0.,.01481
GE
FR 0,0,0,0,46.29
TL 1,3,2,3,-50.
TL 2,3,3,3,-50.
TL 3,3,4,3,-50.
TL 4,3,5,3,-50.
TL 5,3,6,4,-50.
TL 6,4,7,4,-50.
TL 7,4,8,4,-50.
TL 8,4,9,4,-50.
TL 9,4,10,4,-50.
TL10,4,11,5,-50.
TL11,5,12,5,-50.,0.,0.,0.,.02
EX 0,1,3,10,1.
RP 0,37,1,1110,90.,0.,-5.,0.
EN
```

```
CM EXAMPLE 6
CE CYLINDER WITH ATTACHED WIRES.       ALBERTSEN.  CASE 8.
SP0,0,10.,0.,7.3333,0.,0.,38.4
SP0,0,10.,0.,0.,0.,0.,38.4
SP0,0,10.,0.,-7.3333,0.,0.,38.4
GM0,1,0.,0.,30.
SP0,0,6.89,0.,11.,90.,0.,44.88
SP0,0,6.89,0.,-11.,-90.,0.,44.88
GR0,6
SP0,0,0.,0.,11.,90.,0.,44.89
SP0,0,0.,0.,-11,-90.,0.,44.89
GW1,4,0.,0.,11.,0.,0.,23.,.1
GW2,5,10.,0.,0.,27.6,0.,0.,.2
GS0,0,.01
GE?
FR0,1,0,0,465.84
CP1,1,2,1
EX0,1,1,0,1.
RP0,73,1,1000,0.,0.,5.,0.
EX0,2,1,0,1.
XQ
NX
CM SAMPLE PROBLEM NO. 7
CM 1. STRAIGHT WIRE - FREE SPACE
CM 2. STRAIGHT WIRE - PERFECT GROUND
CM 3. STRAIGHT WIRE - FINITELY CONDUCTING GROUND
CE     (SIGMA=.0001   EPSILON=6)
GW0,15,-55.,0.,10.,55.,0.,10.,.01
GE1
FR0,1,0,0,3.
EX1,2,1,0,0.,0.,0.,45.,0.
RP0,2,1,1000,0.,0.,45.,0.
GN1
EX1,1,1,0,45.,0.,0.
RP0,19,1,1000,90.,0.,-10.,0.
GN0,0,0,0,6.,.0001
RP0,19,1,1000,90.,0.,-10.,0.
EN
```

```
CM SAMPLE PROBLEM NO.8
CE STICK MODEL OF AN AIRCRAFT IN FREE SPACE
GW1,1,0.,0.,0.,6.,0.,0.,1.
GW2,6,6.,0.,0.,44.,0.,0.,1.
GW3,4,44.,0.,0.,68.,0.,0.,1.
GW4,6,44.,0.,0.,24.,29.9,0.,1.
GW5,6,44.,0.,0.,24.,-29.9,0.,1.
GW6,2,6.,0.,0.,2.,11.3,0.,1.
GW7,2,6.,0.,0.,2.,-11.3,0.,1.
GW8,2,6.,0.,0.,2.,0.,10.,1.
GE
FR0,1,0,0,3.
EX1,1,1,0,0,.,0.,0.
RP0,1,1,1000,0.,0.,0.
EX1,1,1,0,90.,30.,-90.
RP0,1,1,1000,90.,30.
NX
CM EXAMPLE 9.
CM BISTATIC SCATTERING BY A SPHERE
CM PATCH DATA ARE INPUT FOR A SPHERE OF 1. METER RADIUS
CM THE SPHERE IS THEN SCALED SO THAT KA=FREQ. IN MHZ.
CM THE PATCH MODEL MAY BE USED FOR KA LESS THAN ABOUT 3.
CE FOR THIS RUN, *** KA=2.9.***
SP0,0,.13795,.13795,.98079,78.75,45.,.11975
SP0,0,.51328,.21261,.83147,56.25,22.5,.17025
SP0,0,.21261,.51328,.83147,56.25,67.5,.17025
SP0,0,.80314,.21520,.55557,33.75,15.,.16987
SP0,0,.58794,.58794,.55557,33.75,45.,.16987
SP0,0,.21520,.80314,.55557,33.75,75.,.16987
SP0,0,.96194,.19134,.19509,11.25,11.25,.15028
SP0,0,.81549,.54490,.19509,11.25,33.75,.15028?
SP0,0,.54490,.81549,.19509,11.25,56.25,.15028
SP0,0,.19134,.96194,.19509,11.25,78.75,.15028
GX0,111
GS0,0,47.71465
GE
FR0,1,0,0,2.9
EX1,1,1,0,90.,0.,0.
RP0,19,1,1000,90.,0.,-10.,0.
RP0,1,19,1000,90.,0.,0.,10.
NE0,1,1,11,0.,0.,0.,0.,0.,5.
NE0,1,11,1,0.,0.,0.,0.,5.,0.
NE0,11,1,1,0.,0.,0.,5.,0.,0.
NH0,1,1,11,0.,0.,0.,0.,0.,5.
NH0,1,11,1,0.,0.,0.,0.,5.,0.
NH0,11,1,1,0.,0.,0.,5.,0.,0.
EN
```

```
CM   TEST OF THE NEW NEC WITH SOMMERFELD
CM
CM   GREEN'S FUNCTION FOR RADIAL WIRE SCREEN OVER FINITE GROUND
CM   SCREEN RADIUS=60. M   (ONE WAVELENGTH RADIUS)
CM   SCREEN HEIGHT=.02M    6 RADIAL WIRES
CM
CE
GW0, 12,0.,0.,.02,60.,0.,.02,.003
GR0,6
GE1
FR0,1,0,0,5.
GN2,0,0,0,10.,.01
WG
NX
CM   MONOPOLE ON RADIAL WIRE GROUND SCREEN
CE
GF
GW1,6,0.,0.,.01,0.,0.,15.2,.003
GE1
EX0,1,1,0,1.
RP0,19,2,1001,0.,0.,5.,90.
NX
CM
CM
CM   NOW THE SCREEN JACKED UP TO .6M HEIGHT
CE
GW0,12,0.,0.,.6,30.,0.,.6,.003
GR0,6
GE1
FR0,1,0,0,5
GN2,0,0,0,10,.01
WG
NX
CE  MONOPOLE ON RADIAL GROUND SCREEN
GF
GW1,6,0.,0.,.6,0.,0.,15.6,.03
GE1
EX0,1,1,0,1.
RP0,19,2,1001,0.,0.,5.,90
EN
```

```
CM  TEST OF NEC GROUND/ DIPOLE CLOSE TO GND, MOVING EVER CLOSER
CE
GW0,5,-.25,0.,.1,.25,0.,.1,1.E-10
GE
GN2,0,0,0,4.,-1.7976
EX0,0,3,0,1.
XQ
NX
CE   TEST OF NEC GROUND
GW0,5,-.25,0.,.03,.25,0.,.03,1.E-10
GE
GN2,0,0,0,4.,-1.7976
EX0,0,3,0,1.
XQ
NX
CE   TEST OF NEC GROUND
GW0,5,-.25,0.,.01,.25,0.,.01,1.E-10
GE
GN2,0,0,0,4.,-1.7976
EX0,0,3,0,1.
XQ
NX
CE   TEST OF NEC GROUND
GW0,5,-.25,0.,.003,.25,0.,.003,1.E-10
GE
GN2,0,0,0,4.,-1.7976
EX0,0,3,0,1.
XQ
NX
CE   TEST OF NEC GROUND
GW0,5,-.25,0.,.001,.25,0.,.001,1.E-10
GE
GN2,0,0,0,4.,-1.7976
EX0,0,3,0,1.
XQ
NX
CE   TEST OF NEC GROUND
GW0,5,-.25,0.,.0003,.25,0.,.0003,1.E-10
GE
GN2,0,0,0,4.,-1.7976
EX0,0,3,0,1.
XQ
NX
CE   TEST OF NEC GROUND
GW0,5,-.25,0.,.0001,.25,0.,.0001,1.E-10
GE
GN2,0,0,0,4.,-1.7976
EX0,0,3,0,1.
XQ
EN
```

```
CE   TEST OF NEC GROUND
GW0,5,-.25,0.,.00003,.25,0.,.00003,1.E-10
GE
GN2,0,0,0,4.,-1.7976
EX0,0,3,0,1.
XQ
NX
CE   TEST OF NEC GROUND
GW0,5,-.25,0.,.00001,.25,0.,.00001,1.E-10
GE
GN2,0,0,0,4.,-1.7976
EX0,0,3,0,1.
XQ
NX
CE   TEST OF NEC GROUND
GW0,5,-.25,0.,3.E-6,.25,0.,3.E-6,1.E-10
GE
GN2,0,0,0,4.,-1.7976
EX0,0,3,0,1.
XQ
NX
CE   TEST OF NEC GROUND
GW0,5,-.25,0.,1.E-6,.25,0.,1.E-6,1.E-10
GE
GN2,0,0,0,4.,-1.7976
EX0,0,3,0,1.
XQ
NX
CE   TEST OF NEC GROUND
GW0,5,-.25,0.,3.E-7,.25,0.,3.E-7,1.E-10
GE
GN2,0,0,0,4.,-1.7976
EX0,0,3,0,1.
XQ
NX
CE   TEST OF NEC GROUND
GW0,5,-.25,0.,1.E-7,.25,0.,1.E-7,1.E-10
GE
GN2,0,0,0,4.,-1.7976
EX0,0,3,0,1.
XQ
EN
```

```
CM   TEST OF NEC3 GROUND/BURIED WIRES & SUCH STUFF
CM   1.  DIPOLE IN DIELECTRIC AND LOSSY MEDIA
CM         TEST OF UM CARD
CE
GW 1,7,0,0,-1.5,0,0,1.5,.01
GE
FR 0,0,0,0,10
EX 0,1,4
UM 0,0,0,0,16
RP 0,10,1,1000,0,0,10,0
UM 0,0,0,0,16,-16
RP 0,10,1,1000,0,0,10,0
NX
CM  2. MONOPOLE ANTENNA ON A GROUND STAKE
CE
GW 1,8,0,0,-2.,0,0,0,.01
GW 2,10,0,0,0,0,0,15,.01
GE
FR 0,0,0,0,5
GN 2,0,0,0,10,.01
EX 0,2,1
RP 0,19,2,1001,0,0,5,90
NE 0,1,1,21,5000,0,0,0,0,10
NX
CM  3. MONOPOLE ON 6-WIRE RADIAL GND SCRN
CM       FIRST THE WIRE GROUND SCREEN
CE
GW 1,14,12,0,-.05,.8,0,-.05,.01
GW 1,1,.8,0,-.05,0,0,0,.01
GR 0,6
GE
FR 0,0,0,0,5
GN 2,0,0,0,10,.01
WG
NX
CE      NOW THE MONOPOLE
GF
GW 2,10,0,0,0,0,0,15,.01
GE
EX 0,2,1
RP 0,19,2,1001,0,0,5,90
NE 0,1,1,21,5000,0,0,0,0,10
EN
```

LIST OF REFERENCES

1. G.J. Burke and A. J. Poggio, " Numerical Electromagnetics Code (NEC)-Method of Moments, Part 1: Program Description- Theory," NOSC TD-116, Naval Ocean Systems Center, San Diego, California, Revised January, 1981.

2. J.C. Logan and J.W. Rockway, "The New MININEC (Version 3): A Mini-Numerical Electromagnetic Code," NOSC TD-938, Naval Ocean Systems Center, San Diego, California, September 1986.

3. Stephan Lamont, "NEC on a PC," Paper presented at the 2nd Annual Review of Progress in Applied Computational Electromagnetics, Monterey, California, March 1986.

4. Timothy O'Hara, "Implementation and Evaluation of Mainframe Dependent Program (NEC3) on a Personal Computer (PC)," Master's Thesis, Naval Postgraduate School, Monterey, California, 1988.

5. AST Research, Inc., *AST Premium 386/33 User's Manual*, 1989.

6. Definicon Systems, Inc., *Definicon DSI-780 Coprocessor Board MS/PC DOS System User's Guide and Installation Manual*, 1986.

7. Lahey Computer Systems, Inc., *F77L-EM/32 Reference Manual*, 1989.

8. MicroWay, Inc., *NDP FORTRAN-386 User's Manual*, 1988.

9. Science Applications International Corporation, *SVS FORTRAN-386 Compiler User's Guide for MS-DOS Version 2.8*, 1988.

10. Silicon Valley Software, Inc., *FORTRAN Reference Manual*, 1988.

11. Lorem P. Meissner and Elliott J. Organick, *FORTRAN 77: Featuring Structure Programming*, Addison-Wesley, Reading, Massachussetts, 1980.

12. Michael Metcalf, *Effective FORTRAN 77*, Oxford University Press, New York, 1985.

13. G. J. Burke and A. J. Poggio, "Numerical Electromagnetics Code (NEC)-Method of Moments, Part III: User's Guide," NOSC TD-116, Naval Ocean Systems Center, San Diego, California, Revised January 1981.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
4.	Richard W. Adler, Code EC/AB Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	11
5.	Michael A. Morgan, Code EC/MW Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
6.	James Cahill Kershner, Wright, & Hagaman 5730 General Washington Highway Alexandria, VA 22312	1
7.	Prof. Al Christman Electrical Engineering Department Grove City College Grove City, PA 16127	1

8. Timothy O'Hara — 1
AIRMICS
115 O'Keefe Building
Georgia Institute of Technology
Atlanta, GA 30320

9. Steven Best — 1
Chu Associates
800 Fesler Street
El Cajon, CA 92020

10. Ross L. Bell — 1
Antenna Products Corporation
101 S.E. 25th Avenue
Mineral Wells, TX 76067

11. Prof. J.K. Breakall, CSSL/EE East — 2
Department of Electrical Engineering
Penn State University
University Park, PA 16802

12. G. Burke — 1
Lawrence Livermore Laboratory
P.O. Box 5504, L-156
Livermore, CA 94550

13. Roger Cox — 1
TELEX-HY Gain
8601 E. Cornhusker Highway
Lincoln, NB 68505

14. Gene Tomer — 1
1691 11th Ave.
San Francisco, CA 94122

15. E. Cummins, Jr. — 1
19020 Quail Valley Blvd.
Gaithersburg, MD 20879

16. Dave Faust — 1
Eyring Research Institute
1455 W. 820 N.
Provo, UT 84601

17. Larry Herzon — 1
P.O. Box 1925
Washington, D.C. 20013

18. J.B. Hatfield — 1
Hatfield and Dawson
4226 Sixth Avenue, N.W.
Seattle, WA 98107

19. P.E. Ljung — 1
Sandkullen
Akersberga
SWEDEN S18492

20. Jim Logan, Code 825 — 1
Naval Ocean Systems Center
271 Catalina Boulevard
San Diego, CA 92152

21. Tom Birbaum — 1
McDonnell Douglas
16761 Via Del Campo Court
San Diego, CA 92127

22. Janet McDonald — 1
Commander
USAISEC/ASB/SET-P
Fort Huachuca, AZ 85613-5300

23. Bill Werner — 1
Andrew California Corporation
2028 Old Middlefield Way
Mountain View, CA 94043

24. Tom King — 1
Kentronics, Inc.
P.O. Box 845
Bristol, TN 37621-0845

25. John Bootle — 1
Antenna Technologies
6 Shields Drive
Bennington, VT 05201

26. Stephan Lamont
MCNC
P.O. Box 12889
Research Triangle Park, NC 27709 — 1

27. Fred Raab
Green Mountain Radio Research
50 Vermont Avenue
Winooski, VT 05404 — 1

28. David J. Pinion
P.O. Box 34054
Bethesda, MD 20817 — 1

29. John Strauch
UNISYS Defense Systems
3880 Murphy Canyon Road
San Diego, CA 92123-4403 — 1

30. Tom Silliman
ERI
108 Market Street
Newburg, IN 47638 — 1

31. Richard O'Conner
Randtron Systems
130 Constitution Drive
Menlo Park, CA 94025 — 1

32. Prof. D.C. Baker
ECE Department
University of Pretoria
Pretoria, South Africa 0001 — 1

33. Richard Tell
6141 Racel Street
Las Vegas, NV 89131 — 1

34. David Petke
27020 Ridge Road
Damascus, MD 20872 — 1

35. James J. Wright 1
1127 Norcova Court
Chesapeake, VA 23320

36. Professor John Powers, Code EC/PO 1
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, CA 93940-5000